Introduction
This page contains a simple step-by-step example for using the Expert System Builder (ESB) series of programs. It is by no means exhaustive and is intended merely as an introduction to using the system. This step-by-step example develops a simple pet shop questionnaire that is able to recommend an appropriate pet for someone based upon their lifestyle and personal preferences. It is not intended to be complete nor does it claim to be accurately predict a suitable pet for you so please don't try and implement it and then blame me.
The tutorial does not focus on the detailed mechanics of driving the program as these are well covered in the associated help files.

Assumptions
To simplify the problem I have assuming that our pet shop only stocks (or is aware of) the following animals:

- Cats
- Dogs
- Rabbits
- Hamsters, Gerbils or Mice
- GoldFish
- Budgies
- Tropical Fish

SECTION 1 - Devising the Questions

Getting Started: Traits
You will have some idea of the solutions your ESB system will be providing (in our case it's the list of animals above). Start by 'brainstorming' the traits that each of your solutions has, identify the aspects of each that will allow you to distinguish them from one another. I came up with the following for our solution set:
Cats - low maintenance, not particularly loyal, relatively clean, allergies, cheap to acquire (often free), can become real characters and members of the family. Can handle them often, cuddly, Live 15yrs+, tendency to not come home (wander or get run over).

Dogs - loyal, obedient, can be trained, are a tie, are not clean, faeces in the garden and elsewhere, require regular walks and exercise, slobber, grooming, rarely free (even mongrels), can become real characters and members of the family, like attention, not good in small homes, not good to leave at home all day if you work, Live 15yrs+

Rabbits - Don't really require exercise, not loyal, do not bond, can't really play with a rabbit - they breed like rabbits (unsurprisingly), will cost some money to acquire, require a hutch, hutch requires cleaning often, can't have around the house (normally), Can handle them, Shorter lifespan, 8yrs.

Hamsters, Gerbils or Mice - Need cage, small, relatively inexpensive, don't bond well, not a family member, no exercise required, short lifespans, Smelly, live outside or in shed

GoldFish - very easy to look after, cheap, forgetful, can't cuddle it, very low maintenance

The above are simply my views (and I'm no expert) and not necessarily a true reflection of the actual behaviours (or peoples perceptions) of these animals. However for the purposes of this exercise we'll accept them as being true.

Normalisation
Looking at the list created above it will be possible to identify characteristics or traits to look for in your domain. I've created the following list of issues to consider from the above (plus a few points that came to me as I was thinking)

- Exercise requirements.
- Companionship (and family membership).
- Affection.
- Handling (and cuddling)
- Cleanliness (issues of faeces, dog hair, cat hair etc)
- Space available (at home and for exercise).
- Cost (grooming, feeding, vets bills, purchase etc.)
- Does anyone in the household have any allergies?
- Commitment - How will the pet fit into your lifestyle - do you work all day?
- Children.
- Other pets.
- Lifespan (the animals and perhaps the owners)
- Personal Preference
- Proximity to major roads and/or other dangers.

Devise User Questions
We're now ready to start devising questions based around the normalised traits identified in the previous step. Considering the first few traits in the above set then appropriate questions might be:

Exercise (Trait #1)
Q1. How fit and active are you?
[A] Very fit, I exercise every day and live for keeping fit.
[B] Fit, I exercise often and enjoy keeping fit
[C] Average, I exercise when I get around to it but can do most activities normally.
[D] Unfit, I don't keep fit at all and get tired easily.
[E] Very unfit, I'm the archetypal couch potato.

For people or answered (d) or (e) we need to consider whether they want to get fit so ask the following:

Q2. You don't consider yourself to be very fit but would you like to be?
[A] Yes.
[B] No.
[C] I don't know.

Q3. Do you prefer to be 'out and about' or to spend your time at home?
[A] I love the outdoors.
[B] I like my home the best.
[C] I don't have a preference.
Companionship (Trait #2)
Q4. Which of the following best describes your home life?
[A] I live with my partner and children.
[B] I live with my partner.
[C] I live alone.
[D] I share a property with others.
[E] Something else.

Affection (Trait #3)
Q5. Is it important that your new pet integrates as part of the family and returns affection
[A] Yes.
[B] No.
[C] I'm not bothered.

Companionship (Trait #4)
Q6. Do you want to be able to pick up and cuddle your pet?
[A] Yes.
[B] No.

Cleanliness (Trait #5)
Q7. How house proud are you (or more importantly) your partner?
[A] I'm very house proud, everything has a place and I like to keep things pristine always.
[B] I'm quite tidy, I like my home to look nice but I'm not a fanatic.
[C] I'm tidy(ish) for most of the time, but not overly phased by it.
[D] I'm not tidy, I like my home to look lived in.
[E] My house is on a par with the inside of a skip (is a dumpster in the US?) etc.....

Hopefully you get the idea. Devise questions that allow you to learn a little about the trait we are looking to examine. Some good questions for the remaining traits might be (this list is not exhaustive).
Q8. How big is the house (or apartment) that you live in?
....
Q9. How would you describe the size of your garden?
....
Q10. Do you live near a park, woodland, or open area where pets are allowed to exercise?
....
Q11. Some animals can be expensive, how much are you prepared to spend purchasing one?
....
Q12. How much are you prepared to spend each month on your pet?
....
Q13. Are you allergic to any of the following?
....
Q14. Do you work?
....
Q15. Is there anyone at home whilst you are at work
....
Q16. Do you have any other pets?
....
Q17. Do you have children that still live at home with you
....
Q18. Which of the following age groups best describe your children's age?
....

etc
User questions are entered on this part of the ESB Question Editor screen.
Devise Knowledge Engineer Questions

The questions devised so far have been aimed at the user, i.e., the person using the system and wanting to be advised. The ESB questionnaire is also presented to the domain expert or knowledge engineer (you normally) when using the ESB Knowledge Acquisition Program. With this program the domain expert populates the system with fuzzy data.

The question presented to the knowledge engineer needs to be asked in a different way. Consider when the domain expert enters data for a dog into our pet shop system (see SECTION 2 later). He needs to consider how each of the available options relates to (adds to or detracts from a dog being the recommended solution). For example, our 1st question would be better phrased as follows (the options remain the same)

Q1. How would this animal be suited to people with the following fitness levels
   a) Very fit, I get exercise every day and live for keeping fit
   b) Fit, I exercise often and enjoy keeping fit
   c) Average, I exercise when I get around to it but can do most activities normally
   d) Unfit, I don’t keep fit at all and get tired easily
   e) Very unfit, I’m the archetypal couch potato

We all know that dogs NEED exercise and so a fit person is more suited to having one and someone that sits at home all day would be less suited. Likewise, Question 2 would be better phrased:

Q2. Asked about improving fitness users answer as follows. How would this pet suit each?
   [A] Yes.
   [B] No.
   [C] I don’t know.

Knowledge Engineer questions are entered on this part of the ESB Question Editor screen.

Question Types

In this section, the term 'record' refers to a possible solution entered by the KE using the ESB KA program. The term 'record' and 'possible solution' should be considered interchangeable.

Question Types are tricky and are down to your judgement as the system designer. The 'Question Type' is set by selecting the 'Multiple' or 'Single' radio button next to the Knowledge and System User Questions. These buttons govern the manner in which the Knowledge Engineer and System User respond to questions in the ESB Knowledge Acquisition (KA) and ESB User Interface (UI) Programs. They are either presented with slider bars or check boxes (as below):

Multiple

Single

So what does that mean and what effect does it have? If you’re ahead of me you’ll realise there are 4 possible combinations of Knowledge Engineer Question (KEQ) and System User Question (UserQ). The combinations are explained below:

1. KEQ = Single and UserQ = Single
   You have decided that only one of the possible answers (for this question) should add to the likelihood of any record being the probable answer and you want the user to give only one positive answer. The KE sets the appropriate answer (for each possible solution) in the ESB KA program. The user makes a selection on the ESB
UI program. All possible solutions in the system that match the users selection have their probability of being the actual solution increased, those that don’t have it decreased.

**Q6. Can you (normally) pick up and cuddle this pet? (KEQ)**

**Q6. Do you want to be able to pick up and cuddle your pet? (UserQ)**

[A] Yes.
[B] No.

2. **KEQ = Multiple and UserQ = Single**
   
   You have decided that one or more of the possible answers (for this question) should add to the likelihood of any record being the probable answer but you still want the user to give only one positive answer. The KE sets (using the slider bars) whether each of the possible answers is positive or negative (for each possible solution) in the ESB KA program. The user makes a single selection on the ESB UI program. The setting made by the KE for the users actual response is then used to increase or decrease the probability of each record in the system being the actual solution. Consider the following:

   **Q1. How would this animal be suited to people with these fitness levels? (KEQ)**

   **Q1. How fit and active are you? (UserQ)**

   [A] Very fit, I exercise every day and live for keeping fit.
   [B] Fit, I exercise often and enjoy keeping fit
   [C] Average, I exercise when I get around to it but can do most activities normally.
   [D] Unfit, I don’t keep fit at all and get tired easily.
   [E] Very unfit, I’m the archetypal couch potato.

   For a DOG the KE would make settings as left. This means that if a user answered [A] then the probability of DOG being the solution is increased by the maximum. If the user answered [E] then probability of DOG being the solution is decreased by the maximum.

   Answers [B] and [D] change the probability by varying degrees. [C] is neutral.

3. **KEQ = Single and UserQ = Multiple**
   
   You have decided that only one of the possible answers (for this question) should add to the likelihood of any record being the probable answer but you want the user to rate (or give a preference) for each of the possible answers.

   None of the questions in our example fall into this category and so I will use a different example. Consider someone looking to buy a used car and an ESB system to find a car for them, a question might be:

   **Q1. What colour most closely matches this particular car? (KEQ)**

   **Q1. What colours of car would you consider buying? (UserQ)**

   [A] Red.
   [C] Blue.
   [D] Black.
   [F] Silver.
   [G] Yellow.
   [H] Other

   The car is (normally) only one colour and so the KE will select one of the options when entering data. The user is presented with a set of sliders and rates how he ‘likes’ each colour. ESB looks at the colour set by the KE, for each record in the system, and then increases/decreases the probability of that record being the solution by the response the user gave for that colour.

4. **KEQ = Multiple and UserQ = Multiple**
   
   Use this option when you want the KE to enter values for all options and also the user. The system calculates an increase/decrease in probability for each record (possible solution) based on how closely the users responses match those entered by the KE. Consider Q18 from our example:

   **Q18. What age groups would this animal be suited to? (UserQ)**

   **Q18. Which of the following age groups best describe your childrens age? (UserQ)**

   [A] 0 to 5 years.
[B] 5 to 10 years.
[C] 10 to 15 years.
[D] 15 to 20 years.

**Question Help**

This is straightforward. If you decide that you would like to expand on the exact meaning of a particular question, or give some background information (or explanation) to help the user decide then that is done here. Create an HTML document and reference it from ESB by pressing the 'Add Help' button. You get this dialog to enter the URL.

**Dependencies**

Sometimes you will only want to ask a certain question IF the user's answers to previous questions dictate so. Consider our first two questions:

**Q1. How fit and active are you?**

[A] Very fit, I exercise every day and live for keeping fit.
[B] Fit, I exercise often and enjoy keeping fit
[C] Average, I exercise when I get around to it but can do most activities normally.
[D] Unfit, I don't keep fit at all and get tired easily.
[E] Very unfit, I'm the archetypal couch potato.

**Q2. You don't consider yourself to be very fit but would you like to be?**

[A] Yes.
[B] No.
[C] I don't know.

We only want to ask Q2 if a user responds with [D] or [E]. To do this select the 'Create Reliance' button on the Question Editor program. We want either of these options to lead to Q2 being asked. This is the screen, hopefully it's self explanatory.
It's possible to make a question dependent upon the answers to more than 1 previous question. In our example Q15 is also dependent upon Q14.

**Importance**

When considering the solution to a problem an expert does not assign the same importance to all of the facts he has gleaned about a particular problem. For example, there would be little point in recommending that someone buy a Rolls Royce (or Ferrari) when that person has said that their budget is £20K ($30K). In such a system the budget someone has should have a large bearing on the proposed solution. To assign an importance to questions in our system select one of the radio buttons at the bottom of the screen (default is normal).

For our 'Pet Shop Pet Selector' problem the questions that have most bearing on the final outcome is a matter of judgement (i.e. yours as the KE). I would suggest that Q1 (how active someone is) and Q8 (how much space do they have) are key questions that should have a greater impact. The decision is yours.

**Finally**

When all questions have been created, question types, importance, help and question dependencies assigned then save the system using the 'Save Questions' button.

**SECTION 2 - Populating the Knowledge Base**

This section discusses the development of the knowledge base using the ESB Knowledge Acquisition Program (ESBKA). If you have thought your questions through carefully then this step is relatively straightforward. A number of aspects have already been touched upon in the preceding section.

**Getting Started with ESBKA**

The question file (*.qst) created in the preceding section first needs to be loaded into the ESB Knowledge Acquisition Program. Select the 'Load Questions' button and locate the file you created. Initially the system will contain no records (possible solutions) and so you need to add them (in our case it’s cat, dog rabbit etc...). This is covered below.

**Adding Records (Possible Solutions)**

Returning to our example of the 'Pet Shop Pet Selector' we have already decided that our possible solutions are the following:
Select the 'New Record' button. You will be presented with the following dialog. Type the name of the record we want to create (in our case DOG) and select OK.

![Record Name Dialog]

You will now be presented with the first question. Remember you are answering the question as the domain expert (or KE) as discussed in Question Types above.

These are the settings that the KE makes (for Q1) for a dog in our system.

For a DOG the KE would make settings as left. This means that if a user answered [A] then the probability of DOG being the solution is increased by the maximum. If the user answered [E] then probability of DOG being the solution is decreased by the maximum. Answers [B] and [D] change the probability by varying degrees. [C] is neutral.

Answer the remaining questions in the system for the record DOG.
When complete create the next record (ie CAT) and repeat the process. Save the system as you progress by pressing the 'Save Records' button.

**Describing Records**
The description that you provide for a record in the system is the URL to which the user of the system is directed when that record is selected in the User Interface Program.

**Modifying Records**
There will be occasions (especially when testing) that you need to modify the settings for a particular record so that the system gives the correct result (you can consider this as fine tuning the system). In order to modify a record you first need to load it into ESBKA. From the 'Record Name' drop down list select the record you wish to modify and press 'Load Record'. You can now modify the setting as desired. Remember to save your changes when complete.

**Adding Duplicate Records (Possible Solutions)**
Sometimes you will want the system to make the same recommendation but via a different set of response from the user (i.e. via a different path through the questions). This is achieved by creating a new record in the system (ie DOG 2) and linking it to the same description (see Describing Records below).
Modifying the Question Set (after adding records)
There may be occasions when you realise that the current question set needs to be modified. The problem is that records (solutions) already in the system will have no 'knowledge' of the new question. In order to add a new question, to an existing system, open the file in ESBQE and navigate to the position in the question file that you wish to insert the new question (using the 'Prev' and 'Next' buttons).
From the 'Edit Menu' select 'Insert Question'. Be sure to use this method even if you want to add a question at the end of the existing question set. The ESBQE program presents the following warning, press Yes and complete the new question.

```
Confirmation Required

WARNING: A data file created with the current question set already exists. The records in this data file will be updated with a neutral response by ESBQE. You should use ESBKA to update all existing records with a response for this new question. Do you still want to insert a new question?

[Yes] [No]
```

IMPORTANT: For records entered into the system prior to the new question being added you need to reload those records (in ESBKA) and provide settings for the new question.

SECTION 3 - Testing the System
This section is yet to be completed.

Please Note
There is a lot of help material relating to ESB4. More importantly I don't have time to answer everyones questions. Hopefully this tutorial will help. If you mail me with a question that's answered in the help (or here) you'll be ignored (apologies).

http://www.expertise2go.com/webesie/tutorials/ESIntro/

Introduction to Expert Systems [1]
This tutorial shows you how a computer-based expert system emulates the behavior of a human advisor, presents terminology unique to the field and introduces the activities that must be accomplished to build expert systems.

Good mornin' this is Ace Auto Repair
Hey Ace, this is Sam Peterson. My car wouldn't start this morning and I need some help...

To introduce terms like expert and expertise as they are relevant to expert systems, let's suppose you have been unable to start your car to go to work and have returned to the house to call your favorite mechanic. The dialog might continue something like this...
What happens when you turn the key in the ignition to try to start the car? It turns over OK, but it just won't start.

Here is the beginning of the diagnostic telephone "interview" with your mechanic...

Hmm... are you sure that you aren't out of gas? Well, now that you mention it - I'm not certain the tank is empty, but it probably is.

Based on your input that the starter operates, your mechanic can abandon a number of hypotheses related to electrical problems. Now the expert is evaluating another possible explanation...

As you crank the starter, do you smell gas? No, I turned it over for a long time, but didn't smell anything.

At this point, your mechanic is attempting to confirm the new hypothesis...

Based on what you've told

Thanks for the advice. Mind telling
me, I'm almost certain your car is out of gas. me how you reached your conclusion?

Your mechanic now has enough evidence to diagnose the problem. Once you've heard the recommendation, you might want an explanation of how the conclusion was obtained...

**Introduction to Expert Systems [6]**

When a car won't start my initial suspicion is that the battery is dead, the starter has failed or some other electrical problem exists. Your input that the starter operates makes it more likely that no fuel is getting to the engine. Although you are not sure that the gas tank is empty, the fact that you don't smell gas when the engine turns over supports my conclusion that you are out of gas.

You solved your automotive problem by consulting with an expert. Let's take a look at the definition of expertise relevant to expert systems and the attributes of an effective consultant a computer will have to emulate to substitute for a human advisor...

**Experts, expertise, and consulting** [7]

What's an expert?

An expert is one who possesses specialized skill, experience, and knowledge that most people do not have along with the ability to apply this knowledge using tricks, shortcuts, and rules-of-thumb to resolve a problem efficiently [Harmon and King, 1985]. An expert's advice has to be good enough most of the time for the expert to keep his or her reputation, but is not expected to be perfect or even the globally best available to be considered useful [Hayes-Roth, 1983].

**What are the attributes of effective consultants and consulting?**

- Consulting is goal oriented
- A good consultant is efficient
- A consultation is adaptive
- Consultants are able to work with imperfect information
- Good consultants justify their recommendations by explaining their reasoning

Here's an illustration of each of these attributes from the auto diagnosis example...
Consulting is **goal oriented** [8]

What happens when you turn the key in the ignition to try to start the car?  
It turns over OK, but it just won't start.

The objective in calling your mechanic is to get a very specific answer to a very specific question. You aren't interested in learning how a fuel injection system works or how to rebuild a starter -- even though your expert would be quite capable of providing this information. The objective of the consultation represents a **goal** in expert system terminology, and there can be one or many goals to be satisfied during a consultation with a human expert or a computer-based expert system.

A good consultant is **efficient** [9]

Hmmm...are you sure that you aren't out of gas?  
Well, now that you mention it - I'm not certain the tank is empty, but it probably is.

Your answer to the mechanic's first question eliminated a large number of possible problems from further consideration. A good consultant will stop asking questions relevant to hypotheses that can be rejected based on evidence at hand. Because you said the starter operates (eliminating battery problems as the likely culprit) it makes no sense to ask you if the headlights light or the horn blows.

A consultation is **adaptive** [10]

As you crank the starter, do you smell gas?  
No, I turned it over for a long time, but didn't smell anything.
When the information needed to make a recommendation isn't available, the expert will try other lines of questioning that will help confirm the hypothesis. You weren't sure the gas tank is empty, so the question about smelling gasoline was posed.

**Consultants are able to work with imperfect information [11]**

Based on what you've told me, I'm almost certain your car is out of gas.

Thanks for the advice. Mind telling me how you reached your conclusion?

You aren't sure the fuel tank is empty, but think it probably is. By combining your less than certain information with the evidence provided by the fact that you don't smell gasoline while the engine turns over, the expert can conclude that you are out of gas with a high degree of certainty.

**Consultants justify their recommendations by explaining their reasoning [12]**

When a car won't start my initial suspicion is that the battery is dead, the starter has failed or some other electrical problem exists. Your input that the starter operates makes it more likely that no fuel is getting to the engine. Although you are not sure that the gas tank is empty, the fact that you don't smell gas when the engine turns over supports my conclusion that you are out of gas.

The application of expertise is not a guessing game. A real expert should be able to explain how evidence was used to evaluate rules-of-thumb to develop recommendations.
Given the nature of the consulting process just described, does it make sense to try to deliver advice without the physical presence of an expert?

**Delivering expertise without the expert's physical presence [13]**

The scenario we just examined used a telephone to provide remote access to an expert mechanic. Books and manuals provide other examples of packaged expertise. Methods for delivering advice without the expert's presence that include a stronger goal orientation include checklists, flowcharts and decision tables:

**AUTO DIAGNOSTIC CHECKLIST**

**SECTION 1**
1. Does the starter operate?
   A. Yes (GO TO SECTION 2)
   B. No (GO TO SECTION 3)

**SECTION 2**

A checklist for diagnosing why a car won't start might begin like this. The branching nature of the problem could result in a complex questionnaire.

**START**
- Starter runs?
  - Yes
  - No (GO TO SECTION 3)

- Fuel in tank?
  - Yes
  - No

**Rule**

<table>
<thead>
<tr>
<th>Rule</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Starter runs?</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Smell gas?</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Dead battery</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Out of gas</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Flooded</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Graphical representations of diagnostic procedures like this flowchart, provide an alternative to complex checklists.**

**Decision tables** can provide procedural guidance for complex problems. Attributes of the problem are listed in the condition stub (yellow) and recommendations or intermediate results in the action stub (green). **Rules** (read vertically) specify the action to take for any combination of conditions.

**Computer-based expert systems [14]**

Are the devices just described effective substitutes for a live consultation? For many advising applications, computer-based expert systems match the performance of a human expert more closely than checklists, flow charts and decision tables. While these devices are certainly goal-oriented they may not compare favorably with live interaction or expert systems when efficiency, adaptivity, use of imperfect information and explanation of reasoning are important.

Expert systems represent a practical application of artificial intelligence (AI) research that has been going on for almost the entire history of general-purpose computing. Much has been learned about how we store knowledge and combine what we know to derive new results and solve problems. Expert systems based on these ideas take many forms. The rule-based system used in the examples on this Web site represents knowledge with of **production rules** -- so named because new facts are produced when a rule is proven true. The next slide shows some sample rules used to diagnose why a car won't start.

**Representing knowledge in rule-based systems [15]**

**RULE 1:**
If the result of switching on the headlights is nothing happens or the result of trying the starter is nothing happens
Then the recommended action is recharge or replace the battery

**RULE 2:**
If the result of trying the starter is the car cranks normally and a gas smell is not present when trying the starter
Then the gas tank is empty with 90% confidence
RULE 3:
If the gas tank is empty
Then the recommended action is refuel the car

RULE 4:
If the result of trying the starter is the car cranks normally and a gas smell is present when trying the starter
Then the recommended action is wait 10 minutes, then restart flooded car

Each rule consists of an if part called the premise or antecedent (shown in blue) and a then part called the consequent or conclusion (shown in green). When the if part is true, the rule is said to fire and the then part is asserted -- it is considered to be a fact.

Rule results are often combined to reach a conclusion. The goal of the auto diagnosis is to find a recommended action: what to do to get the car started. Rule 3 tells what to do if the gas tank is empty and rule 2 could prove that the gas tank is empty. If rule 2 fires, rule 3 will also fire and provide a recommended course of action.

The consequent in rule 2 is asserted with 90% confidence. This means that if the rule's premise is true, we are only 90% certain that the car is out of gas. Our computer-based expert might be willing to accept this level of confidence to fire rule 3 and recommend an action.

Reasoning in rule-based systems [16]
One or both of the following reasoning models are typically implemented in rule-based expert systems:

Forward chaining. This method begins with a set of known facts or attribute values and applies these values to rules that use them in their premise. Any rules that are proven true fire and produce additional facts that are again applied to relevant rules. The process continues until no new facts are produced or a value for the goal is obtained. This approach works well when it is natural to gather multiple facts before trying to draw any conclusions and when there are many possible conclusions to be drawn from the facts.

Backward chaining. An alternative approach begins with a rule that could conclude the goal for the consultation ("what action do you recommend to get my car to start?"), tries to obtain values for the attributes used in the rule's premise, then backtracks through additional rules if necessary to determine a value of the goal attribute. When there are many attributes employed in many rules, the backward chaining mechanism produces a more efficient interview than forward chaining because it will not be necessary to ask the user to input values of all of the facts.

An expert system's reasoning mechanism might employ either or both chaining techniques. Knowledge representation and reasoning mechanisms are combined with a user interface in software that represents the complete expert system shown on the next slide. A demonstration of the forward and backward chaining techniques is provided in the Inference Methods and Uncertainty tutorial.

Components of a rule-based expert system [17]
A typical rule-based expert system integrates:

- A problem-domain-specific knowledge base that stores the encoded knowledge to support one problem domain such as diagnosing why a car won't start. In a rule-based expert system, the knowledge base includes the if-then rules and additional specifications that control the course of the interview.
- An inference engine that implements the reasoning mechanism and controls the interview process. The inference engine might be generalized so that the same software is able to process many different knowledge bases.
- The user interface requests information from the user and outputs intermediate and final results. In some expert systems, input is acquired from additional sources such as data bases and sensors.

An expert system shell consists of a generalized inference engine and user interface designed to work with a knowledge base provided in a specified format. A shell often includes tools that help with the design, development and testing of the knowledge base. With the shell approach, expert systems representing many different problem domains may be developed and delivered with the same software environment. This site's demos have been implemented with WebESIE (the Web-Based Expert System Inference Engine) a proprietary, non-commercial
tool. For lists of shells that are free or available commercially, check the PCAI and Carnegie Mellon links under "Other Resources" on the home page.

**Problem selection and knowledge engineering [18]**

- **What kinds of problems make good candidates for expert systems?** Problems involving *diagnosis* like our "my car won't start" scenario are frequently approached with this technology. In fact, many of the early expert systems research efforts focused on medical diagnosis. Expert systems might support *Planning* activities like developing a marketing strategy for a new product. *Instructional* expert systems can take advantage of the adaptive nature of expert system consultations to diagnose student weaknesses and customize instruction. Systems have also been developed that support *configuration* activities like making sure all the necessary components for a customized equipment installation are shipped together.

- **How are expert systems developed?** *Knowledge engineering* is the process of codifying a human expert's expertise and representing that expertise in the knowledge base. If the expert is not capturing his or her own knowledge, this process is usually based on interviewing techniques.
  
  **Attributes** that define problem structure must be identified and named. The subset of the attributes that will represent the goals for a consultation must also be identified.

  **Rules** can then be constructed. Rules combine attributes into logical expressions in the premise and assign values to attributes in the consequent.

  **Control strategies** such as choosing whether to use forward or backward chaining, and the user interface must then be specified. The objective is to make sure the interaction with the expert system user will be similar to interacting with a human expert.

General advice on picking appropriate initial applications is to pick a problem that an expert could solve in about an hour and that is solved repetitively. It is also important to pick a problem for which access to a human expert is available. Expert systems cannot solve problems that human experts cannot solve.

This concludes our **Expert System Introduction**. If you'd like to learn more about how expert system software works, the [Inference Methods and Uncertainty](#) tutorial is recommended. If you are interested in developing knowledge bases take a look at the [Knowledge Engineering](#) presentation. If you have never run the expert systems provided on this Web site, [Using eXpertise2Go's Knowledge Bases](#) is a good starting point.